Práctica Final: Arduino + Pure Data: convertimos el color en sonido

Este último proyecto va a consistir en que conecteis el **sensor de color TCS34725** que vimos en la Práctica 13: Cómo perc... | Librería CATEDU con **Pure Data** para poder generar diferentes sonidos dependiendo de los valores que lea el sensor.

Vamos a proporcionaros alguna herramienta más para que podáis trabajar con más comodidad y versatilidad con los datos que envies desde Arduino a Pure data, datos que vais a utilizar para controlar parámetros de vuestro patch:

Etiquetas en el Serial:

Anteriormente en la Práctica 10: Primera c... | Librería CATEDU enviamos valores desde Arduino a Pure data utilizando el puerto serial. En aquel caso Arduino detectaba cuando se producía un latido y enviaba un mensaje a Pure data para que con cada latido se produjera un sonido. Lo que hacíamos era controlar el sonido con los latidos de nuestro corazón. En aquel ejercicio en Arduino trabajábamos con **un solo parámetro** y todos los valores que Arduino enviaba Pure data procedían de ese único parámetro, por lo tanto, sabíamos sin necesidad de más información que aquellos datos que llegaban a Pure data correspondían con la detección de un latido.

¿Pero y si quisiéramos enviar datos de varios parámetros, como por ejemplo cuándo se produce un latido y las pulsaciones por minuto? Cuando estos datos llegan a Pure data, ¿cómo sabemos que datos corresponden al latido y cuáles, a las pulsaciones por minuto? Utilizando etiquetas.

¿Os acordáis de que en la Práctica 8: OSC. Open ... | Librería CATEDU recibíamos valores desde varios actuadores diferentes: (botón1, botón2, slider1, pulsador3 ...) y que cada uno de ellos tenía

Copyright 2026

su etiqueta para que supiéramos su procedencia? En Arduino si queremos enviar a Pure data, a través del Serial, los valores de diferentes parámetros, como, por ejemplo, el rojo, azul y verde que lee el sensor de color, tendremos que enviar junto con cada valor una etiqueta que nos permita identificarlos.

¿Y cómo vamos a hacer esto? **Empaquetando** en una línea cada etiqueta con su valor. Crearemos este formato de paquete en Arduino, utilizando dos funciones para imprimir datos en el Serial.

Cada línea sera un paquetito que va a contener una etiqueta y un valor. Por ejemplo, un paquete: "rojo: 140"

En Arduino vamos a utilizar dos funciones para imprimir en el serial: Serial.print() y Serial.println()

La diferencia entre Serial.print() y Serial.println() es que **Serial.println()** realiza un **salto de línea** tras imprimir su contenido en el puerto serial, este salto de línea es lo que vamos a utilizar para "cerrar" nuestros paquetes y que tengan una forma que podamos decodificar fácilmente en **Pude data** utilizando objeto "**route**", si os acordáis vimos este objeto en la Práctica 8: OSC. Open ... | Librería CATEDU y nos permite clasificar valores en función de etiquetas. Vamos a ver ahora con ejemplos que hacen en Arduino las funciones Serial.print() y Serial.println():

```
// The SetUp Function:
void setup() {
   Serial.begin(9600);         // Set's up Serial Communication at certain speed.

}

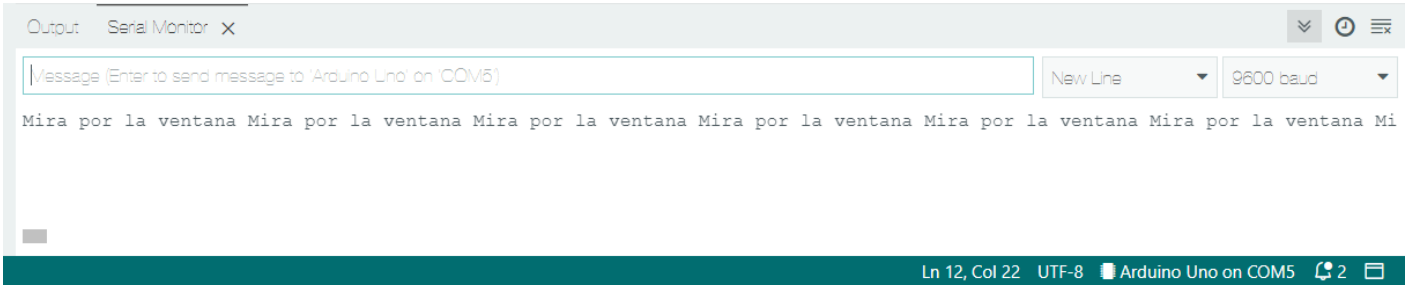
// The Main Loop Function
void loop() {


   Serial.print("Mira ");                    // Send the "Mira" to Serial Plotter.
   Serial.print("por ");                   // Send the "por" to Serial Plotter.
   Serial.print("la ");                   // Send the "la" to Serial Plotter.
   Serial.print("ventana ");                    // Send the "ventana" to Serial Plotter.


delay(10);
```

```
}
```

El código de la ventana superior imprimirá los valores en el serial de la siguiente manera:



La función Serial.print() imprime y se mantiene en la misma línea en la que ha impreso. La función Serial.println() realizará un salto de línea tras imprimir:

```
// The SetUp Function:
void setup() {
   Serial.begin(9600);         // Set's up Serial Communication at certain speed.

}

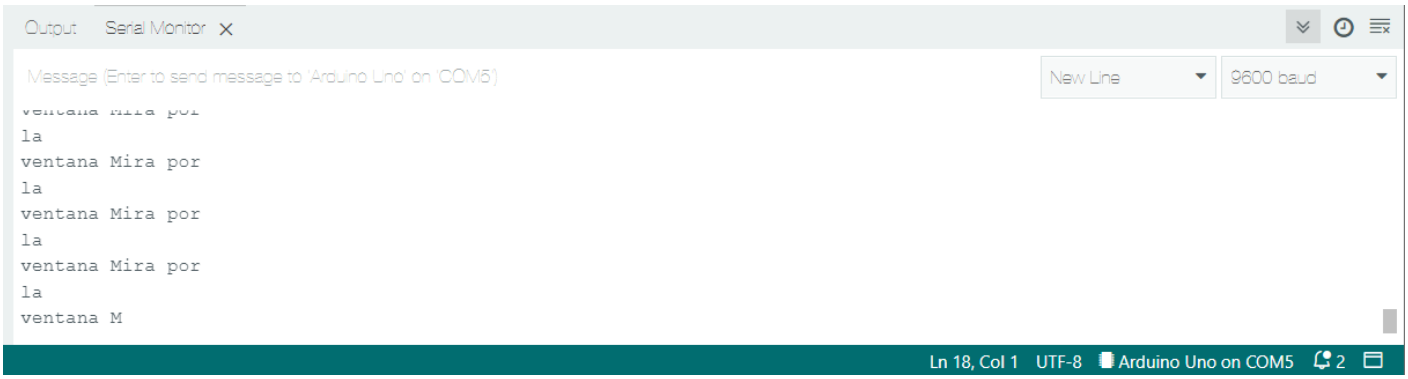
// The Main Loop Function
void loop() {


   Serial.print("Mira ");                    // Send the "Mira" to Serial Plotter.
   Serial.println("por ");                    // Send the "por" to Serial Plotter.
   Serial.println("la ");                    // Send the "la" to Serial Plotter.
   Serial.print("ventana ");                    // Send the "ventana" to Serial Plotter.


delay(10);

}
```

El código de la ventana superior realizará un salto de línea tras imprimir "por " y tras imprimir "la ":



Si queremos que cada frase "Mira por la ventana" se imprima en una sola línea tendremos que colocar la función Serial.println() al final de la frase, imprimiendo la última palabra de esta frase:

```
// The SetUp Function:
void setup() {
   Serial.begin(9600);         // Set's up Serial Communication at certain speed.

}

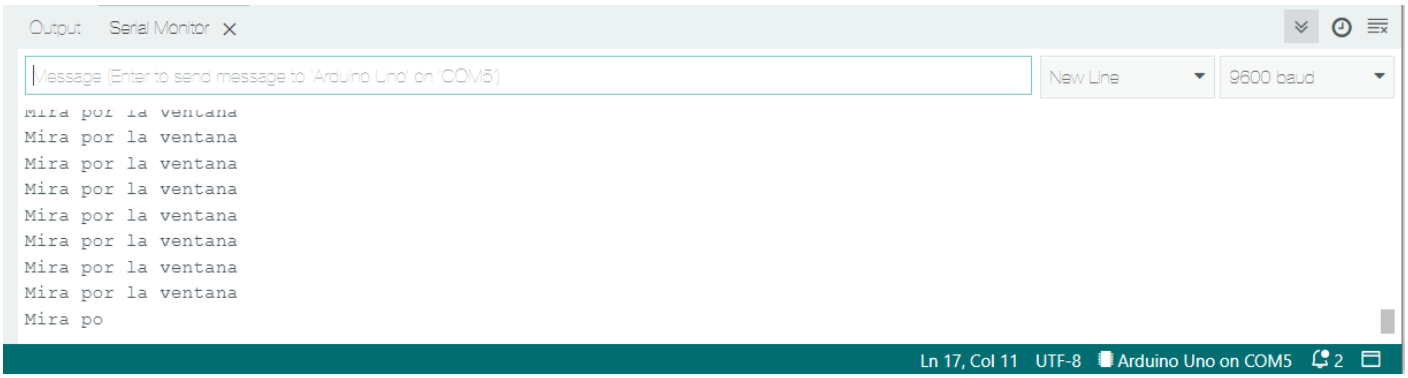
// The Main Loop Function
void loop() {


   Serial.print("Mira ");                    // Send the "Mira" to Serial Plotter.
   Serial.print("por ");                    // Send the "por" to Serial Plotter.
   Serial.print("la ");                    // Send the "la" to Serial Plotter.
   Serial.println("ventana ");                    // Send the "ventana" to Serial Plotter.


delay(10);

}
```

Copyright 2026



Ahora que ya sabemos manejar Serial.print() y Serial.println() vamos a preparar los paquetitos en Arduino para enviarlos a Pure data:

Vamos a crear las etiquetas en Arduino que imprimiremos por el serial junto con los valores correspondientes. Cada **etiqueta** con su **valor** se imprimirá en la **misma línea** y **antes que el valor,** tras la impresión del valor habrá un **salto de línea** que cierre el paquete:

Serial.print("Etiqueta1:"); Serial.println(valor1);

```
18
19  void loop(void) {
20    uint16_t r, g, b, c, colorTemp, lux;
21
22    tcs.getRawData(&r, &g, &b, &c);
23    colorTemp = tcs.calculateColorTemperature(r, g, b);
24    lux = tcs.calculateLux(r, g, b);
25
26    Serial.print("Temperatura-color: "); Serial.println(colorTemp, DEC);
27    Serial.print("Lux: "); Serial.println(lux, DEC);
28    Serial.print("Rojo: "); Serial.println(r, DEC);
29    Serial.print("Verde: "); Serial.println(g, DEC);
30    Serial.print("Azul: "); Serial.println(b, DEC);
31    Serial.print("Clear: "); Serial.println(c, DEC);
32    Serial.println(" ");
33    delay(1000);
34  }
35
```

Figura 1. Lectura de valores del sensor TCS34725 e impresión de esos valores junto con una etiqueta en el puerto serial. (¡Esta imagen solo es una parte del programa!)

Copyright 2026

Cuando lleguen estos datos a través del serial a Pure data utilizaremos el objeto "route" para clasificar los datos recibidos. Los argumentos de este objeto serán las etiquetas que hemos creado en Arduino (figura 1):

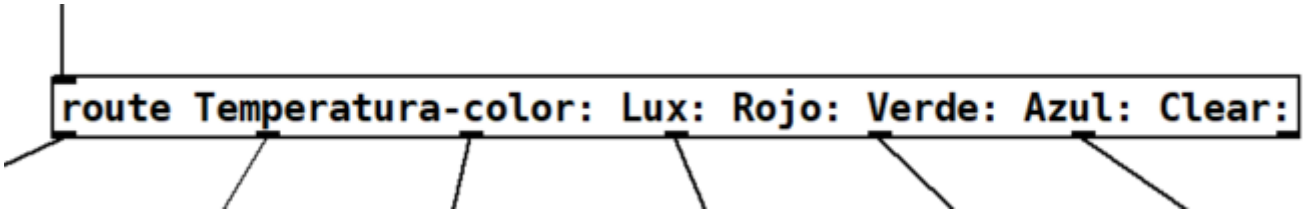


Figura 2. Clasificación en Pure data de los datos enviados por Arduino en la figura X.

Ya sabemos preparar paquetes de datos en Arduino para enviar a traves de Serial!

https://giphy.com/embed/OKPtqN7dlflYcw8gPu

Figura 3. Nosotros poniéndole etiquetas en Arduino a un valor para saber qué contiene.

Mapear valores:

¿Qué es mapear?

Para nostros mapear valores va a ser **ajustar la escala** de un **rango de datos** a otra escala diferente. Vamos a ver un ejemplo:

El sensor TCS34725 con el que estáis trabajando, registra el color utilizando parámetros RGB. En la codificación RGB encontraremos tres valores que nos permitirán definir un color: rojo (r), verde (g), azul (b). El rango de cada uno de estos tres valores es de 0-255. Supongamos que queremos regular el volumen de uno de nuestros instrumentos en función de la cantidad de rojo que tenga un objeto. Como bien sabéis, en Pure data controlamos el volumen modificando la amplitud de las ondas y nuestro rango de volumen ha de estar siempre entre 0 y 1.

Para poder controlar el volumen (0-1) con el parámetro que mide el color rojo (0-255) vamos a tener que adaptar los valores del rojo a los valores del volumen, siendo 0=0 y 255=1. Con esta conversión si el valor del parámetro rojo fuera 127,5 el valor de nuestro volumen sera 0,5.

Para realizar esta conversión vamos a utilizar el objeto "**scale**". Abrid el patch *mapear-scale.pd* para ver cómo funciona.

- En su **primer inlet** recibirá los **valores a mapear**, en nuestro caso los valores del color rojo del objeto que se está evaluando.
- En el **segundo inlet** enviaremos el **valor menor** del rango de los **valores de entrada**, que en nuestro caso es el 0.
- En el **tercer inlet** enviaremos el **valor mayor** del rango de los **valores de entrada**, que en nuestro caso es el 255.
- En el **cuarto inlet** enviaremos el **valor menor** del rango de los **valores de salida**, que en nuestro caso es el 0.
- En el **quinto inlet** enviaremos el **valor mayor** del rango de los **valores de salida**, que en nuestro caso es el 1.

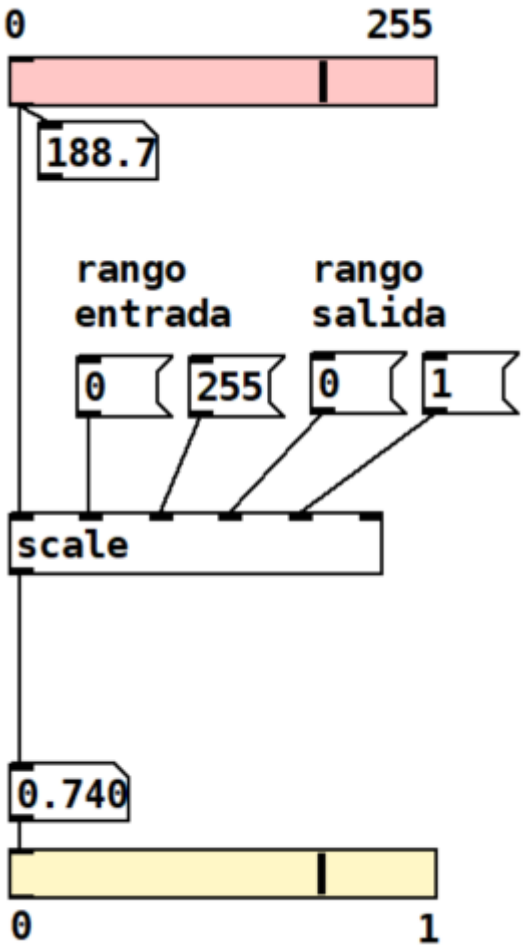


Figura 4. patch *mapear-scale.pd*

Ahora que conocéis el objeto "scale" podréis controlar cualquier parámetro de vuestro patch con los valores que capten los sensores Arduino!!!

> Nos os olvidéis de enviar los mensajes al objeto "scale". Que los mensajes no se envían ellos solos.

Práctica Final

Como os hemos dicho anteriormente este último proyecto va a consistir en que conectéis el **sensor de color** que vimos en la Práctica 13: Cómo perc... | Librería CATEDU con **Pure Data** para poder generar diferentes sonidos dependiendo de los valores que lea el sensor.

Como ves, el enunciado es bastante sencillo, pero para poder hacerlo, es necesario que hayas seguido todos los pasos y actividades que hemos realizado a lo largo del curso, si no, no va a ser muy fácil...

El resultado final tendrá que ser algo parecido a esto:

https://www.youtube.com/embed/LQVEDlo70K0

¿Qué tendremos que entregar?

La entrega tendrá dos partes.

PARTE 1: ARCHIVOS
Esto será una carpeta comprimida en .zip que contenga:
1. Documento de texto con: Nombre completo, explicación de los valores del sensor utilizados, dificultades encontradas.
2. Sketch de Arduino
3. Patch de Pure Data

El nombre de la carpeta será vuestro
ProyectoFinal_Nombre_PrimerApellido_SegundoApellido

PARTE 2: VIDEO CON AUDIO
Será necesario un video con audio de menos de 1 minuto en el que se vea el funcionamiento. El video se llamará: **ProyectoFinalVideo_Nombre_PrimerApellido_SegundoApellido**

Figura 1. Lectura de valores del sensor X e impresión de esos valores junto con una etiqueta en el puerto serial. (¡Esta imagen solo es una parte del programa!)

Figura 2. Clasificación en Pure data de los datos enviados por Arduino en la figura X.

Figura 3. Nosotros poniéndole etiquetas en Arduino a un valor para saber que contiene.

https://giphy.com/gifs/GLSSpain-box-parcel-paquete-OKPtqN7dlflYcw8gPu

Figura 4. patch *mapear-scale.pd*

Revision #1
Created 2023-02-17 11:30:17 CET by Julia del Río
Updated 2023-02-17 11:30:17 CET by Julia del Río

Copyright 2026